ROBOCRITIC: A GROUNDED REWARD MODEL TO ASSESS AND GUIDE ROBOT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable feedback is critical for reinforcement learning-based robot policy improvement and deployment-time guidance. Terminal rewards summarize completed rollouts, while progress estimates based on temporal cues can diverge from the observed state during stalls or regressions. Scalar scores also leave the underlying physical conditions and appropriate next steps implicit. We introduce RoboCritic, a vision-language reward model that can be queried as observations arrive to assess execution and provide grounded guidance. Given a task instruction and one or more camera streams, RoboCritic maintains bounded causal memory with persistent task context and the initial observation, a sliding visual window, and a history of generated feedback. This design retains execution context without storing the full observation history. At each query, it jointly outputs a progress score, a grounded state description, and actionable next-step or corrective feedback. The final query recovers terminal outcome assessment. We train RoboCritic on RPF-1M, extending RoboReward trajectories with over one million aligned process-feedback records for joint supervision. The frozen critic supplies progress-difference rewards for reinforcement learning and language instructions to a frozen vision-language-action policy. Adding state descriptions and guidance as both supervision and history reduces progress-estimation MAE from 0.441 to 0.404. On two real-world manipulation tasks, RoboCritic achieves higher success than the compared baselines in reward-guided policy learning and language guidance of a shared frozen policy.

1 INTRODUCTION

Recent general-purpose robot policies enable language-conditioned manipulation across diverse tasks (Ghosh et al., 2024; Kim et al., 2025b; Black et al., 2025; Physical Intelligence et al., 2025). Further improving these policies and supporting their deployment calls for reliable feedback on task execution. For policy learning, reward signals should capture intermediate progress and setbacks to support credit assignment (Lee et al., 2026; Liang et al., 2026; Chen et al., 2026; Zhang et al., 2025). During execution, feedback can help track task progress, identify failures, and guide the robot's next steps (Dai et al., 2025; Ye et al., 2026). In this work, we study how a general-purpose reward model can provide dense, state-grounded assessments for policy learning and actionable feedback for execution guidance.

Robot reward models commonly summarize task execution with a scalar score, representing either the final outcome or intermediate progress (Lee et al., 2026; Liang et al., 2026; Chen et al., 2026). Such scores provide learning signals for policy optimization, but do not explicitly describe the physical conditions underlying an assessment or how execution should proceed. For example, two partially completed executions may receive the same score but require different actions: transporting a held object or re-grasping a dropped one. Recent work enriches reward supervision with task-stage annotations and procedural reasoning (Chen et al., 2026; Feng et al., 2026). We explore grounded process supervision that pairs progress scores with descriptions of the observed state and next-step or corrective guidance. These complementary targets connect reward prediction to explicit execution context and provide language feedback for guiding robot policies.

We introduce RoboCritic, a general-purpose vision-language reward model that learns to assess and guide robot policies through grounded process supervision. Given a task instruction and visual

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

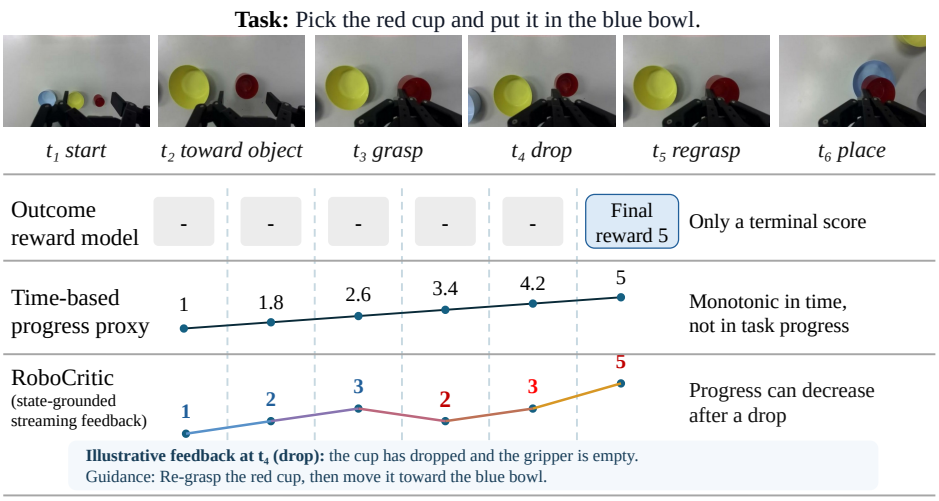


Figure 1: RoboCritic provides dense rewards and grounded guidance for robot policies. Illustrative process scores follow the 1–5 rubric and decrease after a drop. The time proxy increases linearly on the same scale; the example feedback corresponds to t_4.

observations, RoboCritic jointly outputs an ordinal progress score, a structured description of the observed execution state, and next-step or corrective guidance (Figure 1). Following RoboReward, we adopt a shared five-level (1–5) ordinal rubric to maintain a consistent scoring convention across tasks (Lee et al., 2026). We apply this rubric to intermediate execution states, allowing scores to remain unchanged during stalls and decrease after regressions as task progress changes. Joint training with state descriptions and guidance provides explicit supervision of the physical conditions and appropriate next steps associated with each score. The model operates separately from the policy and provides dense assessments at successive observations across tasks.

We construct RPF-1M to provide this aligned supervision, extending the OXE and RoboArena trajectories distributed in RoboReward (Lee et al., 2026; Open X-Embodiment Collaboration et al., 2024; Atreya et al., 2025) with intermediate progress scores, structured state descriptions, and actionable feedback. The progress labels capture achievements and regressions, including within trajectories that ultimately fail. For online assessment, RoboCritic combines recent observations with the task instruction, the initial observation, and a bounded history of its own feedback. This history retains prior assessments and instructions as context for interpreting new observations; each prediction uses only information available up to that point in execution.

The same trained critic supports policy learning and execution guidance. In the feedback-schema comparison, adding state descriptions and guidance as both training targets and historical context reduces MAE from 0.441 to 0.404; state descriptions account for the larger incremental gain. On two real-world manipulation tasks, policies fine-tuned with RoboCritic rewards achieve success rates of 75% and 55%, compared with 45% and 35% using Robometer rewards. When the resulting policy is frozen and shared across guidance methods, RoboCritic achieves 85% human-verified task success over 40 trials, compared with 75% for GPT-5 mini and 60% for Qwen3-VL-4B. These evaluations examine distinct uses of the critic: supplying training rewards and providing instructions during execution.

Our contributions are:

- **Dense reward modeling with grounded feedback.** We introduce RoboCritic, a vision-language reward model that couples online progress assessment with grounded state description and actionable guidance through joint process supervision. A bounded history of generated feedback supports successive assessments from streaming observations.
- **A dataset of aligned process feedback.** We construct RPF-1M, which aligns intermediate robot observations with progress scores, structured state descriptions, and next-step or corrective guidance. The annotation pipeline captures intermediate achievements and regressions, including those within unsuccessful rollouts.

- **Policy learning and execution guidance.** We evaluate progress assessment, feedback-schema and memory ablations, and two downstream uses: reward-guided RL and language guidance of a shared frozen VLA on real robots.

2 Related Work

Reward modeling for robot policies. Visual reward learning draws on goal-conditioned video representations (Ma et al., 2023b;a), pretrained vision-language similarity (Sontakke et al., 2023; Rocamonde et al., 2024), or VLM-generated preferences (Wang et al., 2024). General-purpose models include RoboReward (Lee et al., 2026), which learns five-level outcome assessment, and Robometer (Liang et al., 2026), which combines frame-level progress supervision with trajectory comparisons. GVL (Ma et al., 2025) estimates progress from shuffled frames, ReWiND (Zhang et al., 2025) models regression through video-rewinding augmentation, and Robo-Dopamine (Tan et al., 2026) predicts progress changes between observations. REDS (Kim et al., 2025a) learns subtask-conditioned rewards, while SARM (Chen et al., 2026) jointly predicts task stage and within-stage progress using subtask annotations. DenseReward (Fang et al., 2026) synthesizes failures in simulation for dense reward learning. RoboCritic focuses on general-purpose dense reward modeling through grounded process supervision, aligning progress scores with descriptions of the observed execution state and actionable guidance.

Grounded process supervision and policy guidance. ProgressLM (Zhang et al., 2026a) uses textual or visual references for progress reasoning, while SOLE-R1 (Schroeder et al., 2026) learns temporally grounded reasoning and continuous progress estimates for robot reinforcement learning. ProcVLM (Feng et al., 2026) jointly learns procedure-grounded progress, action segmentation, and future planning. Language feedback also supports closed-loop planning and failure analysis (Huang et al., 2023; Liu et al., 2023; Guo et al., 2024; Duan et al., 2025). RACER (Dai et al., 2025) trains a VLM supervisor and a language-conditioned actor using rich instructions and recovery trajectories, while RoboFAC (Ye et al., 2026) provides failure diagnosis and corrective instructions as an external supervisor for a VLA. We study grounded process supervision for dense reward learning: RPF-1M aligns progress scores, structured physical-state descriptions, and next-step or corrective instructions at sampled observations for joint training. The same trained critic supplies progress-difference rewards for policy optimization and language instructions for a frozen VLA, connecting reward assessment with execution guidance.

Temporal context for reward assessment. Temporal context helps interpret progress when recent observations alone leave earlier completions or regressions ambiguous. R^2VLM (Zhang et al., 2026b) recurrently tracks task decomposition and completion through an evolving chain of thought, while Robo-Dopamine 2.0 (Xu et al., 2026c) conditions pairwise reward prediction on history. StreamingVLM (Xu et al., 2026b) maintains a bounded KV cache with separate visual and textual windows. Following this context design, RoboCritic combines recent observations with a bounded history of generated reward, state, and guidance records to inform later assessments and guidance.

3 Method

RoboCritic learns to assess task progress and provide execution guidance from aligned process supervision. We first define its reward and language outputs, then describe how RPF-1M supplies their training targets. Joint training learns these outputs in a shared model, and feedback history supplies context for online assessment. The trained critic supports robot reinforcement learning and language-guided execution through separate interfaces.

3.1 Grounded Reward Modeling

Given a task instruction i, let $o_t = \{o_t^{(j)}\}_{j=1}^{N_t}$ denote the available synchronized camera views at feedback query t. RoboCritic maps the task and available execution context to a feedback record:

$$\hat{z}_t = (\hat{s}_t, \hat{d}_t, \hat{g}_t) = f_\theta(\mathcal{C}_t), \qquad \hat{s}_t \in \{1, 2, 3, 4, 5\}, \tag{1}$$

where $\mathcal{C}_t$ contains the task instruction, retained observations from $o_{1:t}$, and previous feedback (Section 3.3). Each prediction uses only information available up to the current query. The index t

Table 1: Illustrative process feedback for "Place the brown monkey on the yellow towel." Actionable feedback provides a next-step instruction grounded in the current execution state.

Field	Example output
`reward`	3 (partial progress).
`state_description`	tags: `phase: grasp; motion: grasp; gripper: holding_object; contact: grasped; object: brown monkey;` caption: The gripper holds the brown monkey outside the target placement area on the yellow towel.
`actionable_feedback`	Move the held monkey above the yellow towel.

denotes feedback queries, which need not coincide with low-level control steps; dense assessment means scoring successive intermediate states throughout execution.

Reward assessment. Following RoboReward (Lee et al., 2026), we use a shared 1–5 ordinal rubric to assess the degree of task completion across tasks. We apply it to the current execution state, so $\hat{s}_t$ can remain unchanged during a stall, decrease after a regression, and increase again after recovery. For example, dropping an object can undo progress made by a successful grasp. The same scoring convention applies to intermediate and final states: reading $\hat{s}_T$ at the last query recovers terminal outcome assessment. Changes in these scores supply the RL shaping signal defined in Section 3.4.

Grounded state and guidance. The state description $\hat{d}_t$ records the physical conditions associated with the assessment through structured tags for phase, motion, gripper, contact, and object, together with a short caption. The textual feedback $\hat{g}_t$ explains the score, identifies conditions for further progress or completion, and gives a next-step or corrective instruction. These outputs connect the progress score to the observed execution state and an appropriate continuation. A held object may call for transport toward the target, whereas a dropped object may require re-grasping. Table 1 illustrates the output fields and the instruction component of the feedback.

3.2 Grounded Process Supervision

RPF-1M supplies aligned targets $z_t^* = (s_t^*, d_t^*, g_t^*)$ for reward assessment, state description, and guidance. We construct it from 54,135 OXE and RoboArena trajectories distributed in RoboReward (Lee et al., 2026), retaining task instructions and all available camera views. The annotation pipeline converts rollout-level supervision into feedback at intermediate states, including achievements and regressions within unsuccessful executions (Figure 2).

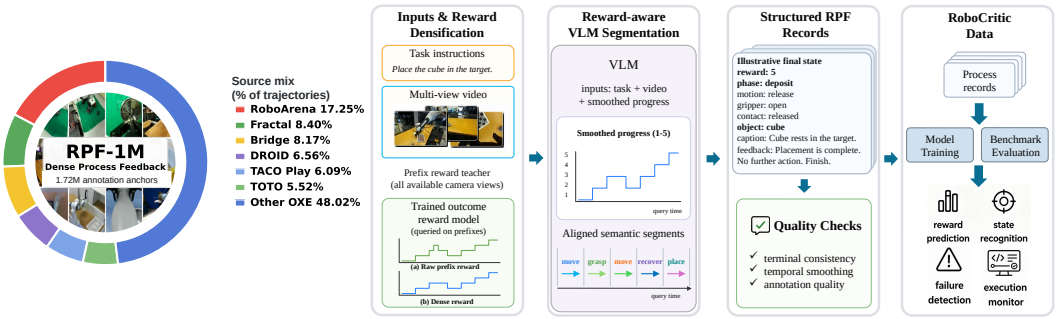
Progress and state annotation. We train a multi-view variant of RoboReward on the RoboReward dataset, initializing it from Qwen3-VL-4B. This model serves as the reward teacher for RPF-1M construction, assigning a 1–5 progress score to each sampled observation prefix. A debounce filter suppresses isolated fluctuations by accepting a score transition only after three consecutive sampled queries support it. The smoothed trace permits decreases after regressions and increases after recovery. Terminal consistency requires only $s_T^* = y_T$, where y_T is the rollout's outcome label; earlier scores may exceed y_T. Score 5 requires visible task completion, and trajectories with inconsistent terminal segments are filtered out. Seed-Pro 2.0 separately annotates structured state tags and captions over semantic execution intervals.

Textual feedback and temporal alignment. We divide the smoothed reward trace into segments with constant scores. For each segment, Seed-Pro 2.0 generates an explanation of the score, conditions for further progress or completion, and a next-step or corrective instruction, using the task and observed execution state. The resulting text is shared across queries within that segment.

Training outputs are scheduled at 0.5 Hz, or one query every two seconds. Let I_k contain the query indices in feedback segment k, with score c_k and textual annotation G_k. At each query, we combine these segment-level targets with the state description aligned to the same timestamp:

$$z_t^* = (c_k, d_t^*, G_k), \qquad t \in I_k. \tag{2}$$

Thus, reward and textual feedback follow reward-change segments, while state descriptions follow semantic execution intervals. Human annotators review held-out records and correct or remove ambiguous examples. Appendix A provides the rubric, dataset statistics, annotation pipeline, and review procedure.



a) Dataset Sources b) Dense Process Feedback Data Engine

Figure 2: **RPF-1M supervision.** Prefix reward scores, state descriptions, and segment-level textual feedback are aligned with training query timestamps to form process-feedback records. The record and progress traces shown are schematic.

3.3 Joint Training and Online Assessment

Joint supervised training. State descriptions and textual feedback serve as joint training targets and, at later queries, as feedback history. The instruction component of the feedback also supports policy guidance. We serialize each target record as $w_t = \text{serialize}(z_t^*)$, ordered as progress score, state tags and caption, and textual feedback. The current state and feedback are decoded after the score and enter the history for subsequent queries. A shared autoregressive decoder learns all fields through the response-generation objective

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(\mathcal{C}_t^*, w_t)\sim\mathcal{D}} \left[\sum_{j=1}^{|w_t|} \log p_\theta(w_{t,j} \mid \mathcal{C}_t^*, w_{t,<j}) \right], \tag{3}$$

where $\mathcal{D}$ is the set of training queries and $\mathcal{C}_t^*$ includes annotated feedback from preceding queries. Prompt tokens are excluded from the loss. The joint objective supervises progress prediction together with the physical state and appropriate next steps associated with each score. We initialize directly from the original Qwen3-VL-4B checkpoint, freeze the vision encoder, and update the language model and visual merger. Appendix B.4 gives the training hyperparameters.

Context for online assessment. Recent observations capture local motion and contact, but may omit earlier events needed to judge current progress. We retain previous feedback as a compact record of earlier assessments and suggested actions. Following StreamingVLM's context design (Xu et al., 2026b), the model combines persistent task context, recent visual observations, and feedback history:

$$\mathcal{C}_t = [\mathcal{P}; \mathcal{V}_t; \mathcal{H}_t], \qquad \mathcal{V}_t = (v_{t-K+1}, \ldots, v_t), \quad \mathcal{H}_t = (\hat{z}_{t-M}, \ldots, \hat{z}_{t-1}), \tag{4}$$

with indices truncated at the start of the trajectory. The persistent context $\mathcal{P}$ retains the task instruction, feedback schema, and initial observation as a stable reference. The visual window $\mathcal{V}_t$ holds the most recent K observation blocks, each containing the available synchronized views with camera identifiers. The history $\mathcal{H}_t$ holds the most recent M generated feedback records. We use $K = 8$ and $M = 32$ by default.

As observations arrive, the visual window advances; after each query, the new feedback record is appended to the history and records beyond its budget are discarded. Current visual evidence allows subsequent predictions to revise prior assessments after progress or regression. Suggested actions in the history are treated as proposals, with their execution assessed from new observations. Training uses annotated history, whereas inference uses the model's own outputs. With fixed visual and response token budgets, the retained context remains bounded independently of trajectory length. Figure 3 illustrates this organization.

3.4 Policy Learning and Execution Guidance

Rewards for reinforcement learning. RoboCritic remains frozen while the robot policy learns from its assessments. At each query, we convert the change in predicted progress into a dense

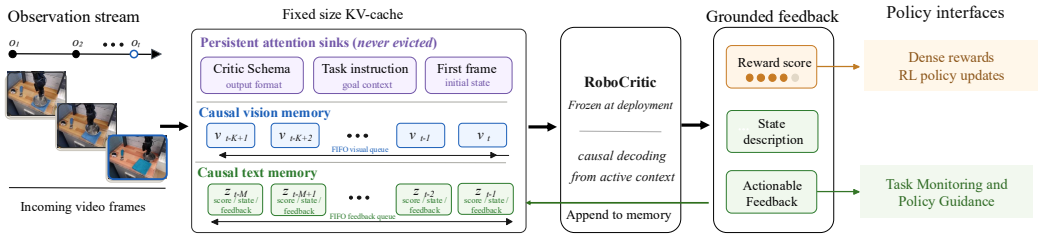


Figure 3: **RoboCritic architecture.** A bounded KV cache combines persistent task context with separate visual and language queues. A shared autoregressive decoder generates progress scores, state descriptions, and actionable feedback, which are appended to the queue for subsequent predictions. Score differences supply RL rewards, while language instructions guide policy execution.

shaping signal:

$$r_t^{\mathrm{RC}} = \hat{s}_t - \hat{s}_{t-1}. \tag{5}$$

Score increases yield positive rewards, unchanged scores yield zero, and decreases yield negative rewards. This assigns equal numerical increments to adjacent ordinal categories as a practical shaping rule. The resulting signal trains the policy through reinforcement learning without updating RoboCritic.

Language guidance for execution. The instruction component of $\hat{g}_t$ guides a frozen VLA π_ϕ through its language interface. We initialize the active instruction with the original task i. At each query, RoboCritic assesses the new observations and generates feedback. The interface interprets this response to retain the active instruction, replace its guidance component with a new subgoal or correction while preserving the task, or terminate on a completion indication such as `finish`. These decisions are derived from the generated text. The VLA acts on the active instruction, and the resulting observations supply the next critic query.

4 Experiments

Our evaluation examines intermediate and terminal reward prediction, state and guidance quality, reward-guided policy learning, execution verification, and VLA guidance. We also compare supervision variants and ablate feedback-history components.

4.1 Experimental Setup

Data and models. We evaluate process feedback on the human-reviewed RPF-1M test split and terminal scores on the RoboReward test set. RPF-1M extends RoboReward's source trajectories with process annotations, so these evaluations assess complementary outputs within related data distributions. PrefixCritic and RoboCritic are both initialized directly from the original Qwen3-VL-4B checkpoint (Bai et al., 2025) and trained on the same RPF-1M trajectories and query timestamps, with a frozen vision encoder. PrefixCritic uses only progress-score supervision, whereas RoboCritic jointly learns progress scores, state descriptions, and textual feedback. At each query, PrefixCritic independently re-encodes eight approximately uniformly sampled timesteps from the observed prefix. RoboCritic combines eight recent visual timesteps with the initial observation and 32 previous feedback records. All available views at each selected timestep are serialized. Their comparison therefore reflects differences in both output supervision and temporal context.

Comparisons and metrics. We compare task-specific reward models, general VLMs, and alternative context designs. Progress is measured by mean absolute error (MAE) on the 1–5 scale and within-trajectory Spearman correlation, denoted value-order correlation (VOC). Time Proxy, $1 + 4t/T$, provides a non-visual reference that increases linearly from 1 to 5. It uses the eventual rollout duration T and is a retrospective control rather than a deployable online baseline when that duration is unknown.

Table 2: Process-score prediction on RPF-1M. K denotes visual timesteps and M language records; the context column describes the accessed history context. Time Proxy uses the eventual rollout duration T.

Method	Context	Overall	RoboArena	OXE
		MAE ↓ / VOC ↑	MAE ↓ / VOC ↑	MAE ↓ / VOC ↑
Time Proxy	t/T	1.256 / 0.783	1.274 / 0.771	1.236 / **0.858**
Robometer	Prefix	0.837 / 0.602	0.702 / 0.537	1.008 / 0.650
LocalWindow	Recent K	0.535 / 0.605	0.601 / 0.419	0.453 / 0.742
PrefixCritic	Strided K	0.451 / 0.715	0.439 / 0.624	0.466 / 0.783
RoboCritic	$K + M$	**0.404** / **0.821**	**0.413** / **0.823**	**0.392** / 0.819

Table 3: Terminal-query ordinal MAE on the RoboReward test set. Overall averages all 2,831 trajectories (1,000 RoboArena and 1,831 OXE).

Method	Overall ↓	RoboArena ↓	OXE ↓
Qwen3-VL Instruct (4B) Bai et al. (2025)	0.998	0.929	1.036
Gemini Robotics-ER 1.5 Gemini Robotics Team (2025)	0.937	1.002	0.902
Gemini 3 Pro (Preview) Google DeepMind (2025)	0.975	1.234	0.833
GPT-5 mini OpenAI (2025)	0.746	0.862	0.683
RoboReward 4B Lee et al. (2026)	0.833	0.806	0.847
RoboReward 8B Lee et al. (2026)	0.698	0.768	0.660
PrefixCritic	0.488	0.621	0.416
RoboCritic	**0.463**	**0.572**	**0.403**

4.2 Assessing Execution Progress

Table 2 evaluates progress predictions at intermediate queries. RoboCritic obtains the lowest overall MAE (0.404) and highest overall VOC (0.821) among the reported methods, compared with 0.451/0.715 for PrefixCritic and 0.535/0.605 for LocalWindow. The comparison with PrefixCritic reflects differences in both supervision and temporal context; it does not isolate the contribution of streaming inference.

Time Proxy attains the highest OXE VOC (0.858) despite a substantially larger MAE (1.236). Only 3% of OXE test trajectories contain a score decrease, compared with 14% for RoboArena, so rank correlation alone provides limited evidence of sensitivity to regressions. On the failure-state subset, RoboCritic achieves 0.307 MAE and 0.816 VOC, compared with 0.401/0.582 for RoboReward (Table 5(b)). This subset includes missed grasps, dropped objects, stalls, and recovery states, where elapsed time alone does not determine progress.

4.3 Terminal Reward Compatibility

We also evaluate the parsed progress score at the terminal query on the RoboReward test set. RoboCritic obtains an overall MAE of 0.463, compared with 0.488 for PrefixCritic and 0.698 for RoboReward 8B (Table 3). This result shows that the same process-feedback interface also supports terminal outcome assessment.

4.4 State Grounding and Execution Guidance

We evaluate state descriptions and actionable feedback on 500 human-reviewed RPF-1M test samples (Table 4). Soft Tag averages relaxed matching accuracies for phase, motion, gripper, and object. A separate VLM judge (GPT-5.6-luna, temperature 0) scores caption grounding and feedback usefulness in $\{0, 0.5, 1\}$ from two frames and reviewed references. Temporal stability measures non-contradiction between adjacent judged queries with unchanged reference phase. Appendix A.5 details matching, query selection, and judging criteria.

State descriptions. RoboCritic achieves a soft tag accuracy of 0.722, compared with 0.552 for Qwen3-VL-4B, and a caption-grounding score of 0.605, compared with 0.358 for GPT-5 mini, the strongest reported baseline on this metric. These results indicate better agreement with the reviewed

Table 4: State and guidance quality, completion verification, and guidance success rate. State and language metrics use human-reviewed RPF-1M test set; grounding and usefulness are judged from images and reviewed references. Guidance SR reports human-verified success over 40 trials per method with a frozen VLA (20 per task). "–" denotes unreported results.

Method	State description		Language guidance		Verification			Guidance SR
	Soft tag	Ground.	Useful.	Stable.	Recall	Prec.	F1	
Qwen3-VL-4B	0.552	0.329	0.419	0.702	0.36	0.61	0.45	60%
GPT-5 mini	0.449	0.358	0.469	0.812	0.42	0.60	0.49	75%
RoboReward	–	–	–	–	0.65	0.53	0.58	–
Robometer	–	–	–	–	0.40	0.55	0.46	–
RoboCritic	**0.722**	**0.605**	**0.744**	**0.965**	**0.85**	**0.81**	**0.83**	**85%**

Table 5: Context ablations (left) and progress prediction on failure/recovery states (right). Removing the full sink removes task and schema context as well as the initial observation.

(a) Memory components.

Variant	MAE ↓	VOC ↑
RoboCritic	**0.404**	**0.821**
w/o process-state mem.	0.432	0.781
w/o first observation	0.443	0.782
w/o persistent sink	0.506	0.712

(b) Failure-state subset.

Method	MAE ↓	VOC ↑
Time Proxy	1.287	0.552
Robometer	0.696	0.485
RoboReward	0.401	0.582
RoboCritic	**0.307**	**0.816**

execution states under the respective evaluation rules. Appendix C.3 reports detailed exact-match accuracy, averaging 50.4% across the five fields, as a complementary measure to soft matching.

Guidance quality. RoboCritic achieves feedback usefulness of 0.744 and temporal stability of 0.965, compared with 0.469 and 0.812 for GPT-5 mini. Thus, its guidance receives higher reference-based usefulness scores and contradicts itself less often within an unchanged annotated phase. The additional execution metrics assess completion verification and the utility of guidance for a language-based policy.

Execution verification and guidance. Table 4 also reports execution-feedback metrics. Recall, precision, and F1 measure completion classification on RoboArena videos, where a verifier uses predicted progress and state descriptions to mark completion. RoboCritic obtains an F1 score of 0.83, compared with 0.58 for RoboReward. Guidance SR measures human-verified task success over 40 trials per method (20 each for pick-and-place and insertion), using the same VLA checkpoint fine-tuned with RoboCritic rewards and frozen across guidance methods. Each trial has a two-minute time limit. RoboCritic guidance achieves 85% task success, compared with 75% for GPT-5 mini and 60% for Qwen3-VL-4B, gains of 10 and 25 percentage points, respectively. These results support the execution utility of the learned feedback in the evaluated setting. Appendix D.2 details the guidance interface and evaluation protocol.

4.5 Effect of State Descriptions and Language Guidance

Table 6 compares three feedback schemas used as both training targets and historical context. All variants share the initialization checkpoint, training data, training steps, visual window ($K = 8$), and history length ($M = 32$). Each variant retains only the fields it is trained to predict.

Table 6: Effect of state descriptions and language guidance used as training targets and historical context.

Output fields	MAE ↓	VOC ↑
Score only	0.441	0.732
Score + state	0.413	0.801
Score + state + guidance	**0.404**	**0.821**

Adding state descriptions reduces MAE from 0.441 to 0.413 and increases VOC from 0.732 to 0.801. Including guidance further improves these metrics to 0.404 and 0.821. State descriptions provide the larger incremental gain, while guidance offers an additional improvement. These results measure the combined contribution of the added fields as training supervision and feedback history.



(a) Real-robot setups

Reward source	Pick Place SR ↑	Insertion SR ↑
Base Policy	20%	10%
Human Reward	50%	45%
RoboReward	40%	30%
Robometer	45%	35%
RoboCritic	**75%**	**55%**

(b) RL success rates

Figure 4: Real-robot reward-guided policy learning. (a) Pick-and-place (left) and insertion (right) setups. (b) Success rates with different reward sources over 20 evaluation trials per method and task.

4.6 Ablatoins of Feedback History

Table 5(a) examines components of the streaming context. Removing feedback history increases MAE from 0.404 to 0.432 and decreases VOC from 0.821 to 0.781. Removing the initial observation increases MAE to 0.443. Removing the entire persistent sink gives 0.506 MAE, but also removes the task and schema context, so this intervention does not isolate visual memory alone. These ablations support the utility of retained context for progress estimation; they do not measure its contribution to complete-task execution. Appendix C.4 reports visual- and feedback-memory budget sweeps. All headline streaming predictions use generated feedback history at evaluation time.

4.7 Reward-Guided Policy Learning

Policy learning. We use an AgileX Piper with external and wrist cameras (Figure 4(a)). A $\pi_{0.5}$ (Physical Intelligence et al., 2025) policy trained from 25 demonstrations per task provides the common initialization for RL-Token (Xu et al., 2026a) fine-tuning on pick-and-place and insertion. All reward-based fine-tuning methods use the same RL procedure and a budget of 50 training rollouts per task, with reward queries at 2 Hz. Each method is evaluated over 20 trials per task. Human Reward uses manually annotated 1–5 progress scores and the same adjacent-score differencing as RoboCritic (Appendix D.1).

RoboCritic-guided fine-tuning achieves 75% and 55% success on the two tasks, compared with 45% and 35% for Robometer and 40% and 30% for RoboReward (Figure 4(b)). The gains over Robometer are 30 and 20 percentage points, respectively, under the reported evaluation. These results support the utility of RoboCritic-based rewards for policy learning under a matched rollout budget. The experiments do not compare progress-difference shaping with terminal-only rewards from the same model.

5 Conclusion and Limitations

We introduced RoboCritic, a vision-language reward model that jointly predicts progress scores, grounded state descriptions, and actionable guidance from streaming observations. Trained on aligned process feedback in RPF-1M, it uses bounded causal memory to assess advances, stalls, and regressions during execution. The same critic supplies progress-difference rewards for policy learning and language instructions to a frozen policy. Experiments show improved progress estimation and higher success rates than the evaluated baselines in reward-guided learning and frozen-policy guidance on two real-world manipulation tasks.

RoboCritic has several limitations. VLM-assisted process annotation is costly to scale and can introduce bias. The 1–5 rubric is compact but ambiguous for intermediate states, complicating consistent annotation across tasks and embodiments. Generated feedback histories may propagate prediction errors. Scaling requires better annotation protocols and broader task, embodiment, and viewpoint coverage, particularly egocentric observations. Guidance is evaluated with a shared policy trained using RoboCritic rewards; transfer to other policies remains untested. Finally, using the state and guidance outputs as richer RL signals for subgoal shaping and recovery remains future work.

AI USE STATEMENT

In this work, we used a vision–language model (Seed-Pro 2.0) to construct process-feedback annotations for RPF-1M, including semantic stage segmentation, structured state tags, captions, and actionable feedback, as described in Section 3 and Appendix A. Source videos come from existing robot datasets. VLM-generated annotations are filtered for terminal consistency ($s_T = y_T$), and human annotators review the test records. We also used generative AI assistants to discuss candidate experimental designs and evaluation protocols. The authors retain responsibility for selecting and validating the final protocols, verifying numerical results, and drawing scientific conclusions. Additionally, we used generative AI writing and coding assistants to draft and edit manuscript text, improve readability, format LaTeX tables and references, and write plotting scripts for some figures. All AI-assisted text, code, claims, and figures were reviewed by the authors. We take responsibility for all final content, including text, claims, and artifacts produced with generative AI.

REPRODUCIBILITY STATEMENT

The appendix documents the dataset construction, model configuration, and evaluation settings used in the reported experiments. Appendix A describes RPF-1M, including the ordinal rubric, tag schema, $s_T = y_T$ filtering, and auto vs. human-polished agreement. Appendix B describes streaming memory, training, baselines (including Time Proxy), and metrics. Appendix C reports K/M sweeps, feedback-history ablations, tag accuracy, and qualitative traces. Appendix D describes ordinal RL shaping and VLA guidance.

REFERENCES

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Cyrus Neary, Edward S. Hu, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Tony Wang, Xingfang Yuan, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martín-Martín, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. RoboArena: Distributed real-world evaluation of generalist robot policies. In *CoRL*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *RSS*, 2025. doi: 10.15607/RSS.2025.XXI.010.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1c92edb990a05f2269f0cc3afbb4c952-Abstract-Conference.html.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. RACER: Rich language-guided failure recovery policies for imitation learning. In *ICRA*, 2025. URL https://rich-language-failure-recovery.github.io/.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/70a06501001e1820fd1eb9ee821302d2-Abstract-Conference.html.

Yu Fang, Wanxi Dong, Jiaqi Liu, Yue Yang, Mingxiao Huo, Yao Mu, Huaxiu Yao, Li Erran Li, Daniel Szafir, and Mingyu Ding. DenseReward: Dense reward learning via failure synthesis for robotic manipulation. *arXiv preprint arXiv:2607.13033*, 2026. URL https://arxiv.org/abs/2607.13033.

Youhe Feng, Hansen Shi, Haoyang Li, Xinlei Guo, Yang Wang, Chengyang Zhang, Jinkai Zhang, Xiaohan Zhang, Jie Tang, and Jing Zhang. ProcVLM: Learning procedure-grounded progress rewards for robotic manipulation. *arXiv preprint arXiv:2605.08774*, 2026. URL https://arxiv.org/abs/2605.08774.

Gemini Robotics Team. Gemini Robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *RSS*, 2024. doi: 10.15607/RSS.2024.XX.090.

Google DeepMind. Gemini 3 Pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf, 2025.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding language model by detecting and recovering from plan-execution misalignment. In *IROS*, 2024. URL https://sites.google.com/view/doremi-paper.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782, 2023. URL https://proceedings.mlr.press/v205/huang23c.html.

Changyeon Kim, Minho Heo, Doohyun Lee, Honglak Lee, Jinwoo Shin, Joseph J. Lim, and Kimin Lee. Subtask-aware visual reward learning from segmented demonstrations. In *ICLR*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/277628cff838927d869cd1f671328ce0-Abstract-Conference.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Bıyık, and Jesse Zhang. RoboMeter: Scaling general-purpose robotic reward models via trajectory comparisons. In *RSS*, 2026.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3468–3484, 2023. URL https://proceedings.mlr.press/v229/liu23g.html.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *ICML*, 2023a. URL https://proceedings.mlr.press/v202/ma23b.html.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *ICLR*, 2023b.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/54854cf15a24fff9f5134a8641136fe4-Abstract-Conference.html.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024.

OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *ICLR*, 2024.

Philip Schroeder, Thomas Weng, Karl Schmeckpeper, Eric Rosen, Stephen Hart, and Ondrej Biza. SOLE-R1: Video-language reasoning as the sole reward for on-robot reinforcement learning. *arXiv preprint arXiv:2603.28730*, 2026. URL https://arxiv.org/abs/2603.28730.

Sumedh A. Sontakke, Jesse Zhang, Sebastien M. R. Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *NeurIPS*, 2023.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Cheng Chi, Yuheng Ji, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. General process reward modeling for robotic reinforcement learning. In *CVPR*, pp. 22412–22422, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Tan_General_Process_Reward_Modeling_for_Robotic_Reinforcement_Learning_CVPR_2026_paper.html.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-F: Reinforcement learning from vision language foundation model feedback. In *ICML*, 2024.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL Token: Bootstrapping online RL with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026a.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. StreamingVLM: Real-time understanding for infinite video streams. In *ICLR*, 2026b.

Yijie Xu, Haopeng Jin, Run Zhou, Shengbang Liu, Sixiang Chen, Hongyang Cheng, Sicheng Hu, Peterson Co, Jinwen Luo, Huajie Tan, and Shanghang Zhang. Robo-Dopamine 2.0: History-conditioned and OOD-aware process reward modeling for robotic manipulation. *arXiv preprint arXiv:2608.15680*, 2026c. URL https://arxiv.org/abs/2608.15680.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. RoboFAC: A comprehensive framework for robotic failure analysis and correction. In *WAICA*, 2026. URL https://arxiv.org/abs/2505.12224v4.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Bıyık, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations. In *CoRL*, 2025. URL https://openreview.net/forum?id=XjjXLxfPou.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. ProgressLM: Towards progress reasoning in vision-language models. In *ACL*, pp. 11243–11271, 2026a. doi: 10.18653/v1/2026.acl-long.516. URL https://aclanthology.org/2026.acl-long.516/.

Yuelin Zhang, Sijie Cheng, Chen Li, Zongzhao Li, Yuxin Huang, Yang Liu, and Wenbing Huang. Recurrent reasoning with vision-language models for estimating long-horizon embodied task progress. In *CVPR*, pp. 41150–41159, 2026b. URL https://github.com/GLAD-RUC/R2VLM.

Appendix overview. The supplementary material is organized as follows. Appendix A describes RPF-1M: ordinal rubric, tag schema, dataset statistics, labeling with $s_T = y_T$ filtering, auto vs. human-polished agreement, and language evaluation. Appendix B provides streaming memory, training, baselines (including Time Proxy), and metrics. Appendix C reports K/M sweeps, feedback-history ablations, tag accuracy, and qualitative traces. Appendix D describes ordinal RL shaping and VLA guidance.

A RPF-1M Construction

RPF-1M converts terminal rollout supervision into per-query process feedback. Each labeled record contains a task instruction, the available camera observations at the query state, a progress score, a structured state description, and actionable feedback. This section describes the feedback format, dataset composition, dense annotation pipeline, and human verification procedure.

A.1 Feedback Definition

For a task instruction i and an observation prefix $o_{\leq t}$, the target feedback record is

$$z_t = (s_t, d_t, g_t), \qquad s_t \in \{1, 2, 3, 4, 5\}, \tag{6}$$

where s_t is the progress score, d_t is a grounded state description, and g_t is textual feedback containing a score explanation, conditions for further progress or completion, and a next-step or corrective instruction. The same record format is used for training and evaluation. Terminal outcome prediction is recovered by querying the record at the final timestep and reading the reward field. The reward follows the RoboReward 1–5 convention, but it is assigned to the current execution state rather than to the full rollout outcome. The score is based on task-relevant visual state, so it may stay constant during pauses and may decrease after regressions such as object drops, wrong-object interactions, or failed insertions. Table 7 summarizes the rubric.

Table 7: Compact ordinal process-score rubric. The score evaluates the current task state, not normalized time. Bins are an interface, not an equidistant reward.

Score	State-level criterion	Typical example
1	No meaningful task progress, wrong-object interaction, severe failure, or idle behavior unrelated to the instruction.	The gripper moves near the wrong object, the object is dropped out of reach, or the robot remains idle.
2	Early task-relevant progress, but without a reliable useful interaction.	The gripper approaches the correct object or begins aligning with the target, but no stable contact has been made.
3	Partial progress: a meaningful task-relevant interaction has occurred, while the main spatial or contact goal remains unsatisfied or unstable.	The robot touches or grasps the correct object, but it has not transported, inserted, or placed it correctly.
4	Near completion: most instruction-relevant conditions are satisfied, but a minor alignment, release, contact, stability, or placement condition remains.	The object is near the target container or slot, but is not yet released, stable, or fully inserted.
5	Complete success: the current state satisfies the instruction-relevant object, spatial, contact, and release conditions.	The correct object is released in the target container, or the inserted object is seated in the target fixture.

Table 8: Structured fields used in the RPF-1M state description.

Field	Description	Common values
`phase`	High-level execution stage.	`approach`, `grasp`, `transport`, `align`, `release`, `recover`, `complete`, `fail`, `idle`.
`motion`	Visible motion primitive.	`grasp`, `lift`, `insert`, `place`, `release`, `none`, `other`.
`gripper`	Visible gripper state.	`open`, `closing`, `holding_object`, `empty`, `occluded`.
`contact`	Visible object contact.	`none`, `near`, `touching`, `grasped`, `released`.
`object`	Task-relevant object.	Object name from the instruction or visible scene, or `null`.
`caption`	Human-readable rendering of the same state.	Free-form grounded text.

Table 9: RPF-1M dataset statistics by source (aggregated over all splits). "Frames" denotes dense timestamped annotation anchors, not raw video frames.

Source	Episodes	Frames	Views
RoboArena	9,337	602,506	2–3
Fractal / RT-style OXE	4,549	30,303	1
Bridge / TACO	7,721	63,348	1–2
Other OXE	32,528	1,027,146	1–3
Total	54,135	1,723,303	1–3

A.2 State description and feedback format

Each state description combines structured tags with a short natural-language caption, and each feedback record additionally includes actionable feedback. Table 8 lists the structured fields in d_t.

The tags `phase`, `motion`, `gripper`, `contact`, and `object` in Table 8 are the compact process-state record stored in memory. The `caption` is a human-readable rendering of the same state and must be supported by visible evidence. Textual feedback g_t explains why the current execution state merits its score, identifies conditions needed for further progress or completion, and gives a high-level next-step or corrective instruction (e.g., re-grasp, align, transport, or release). The instruction is expressed through the VLA's language interface rather than as joint commands.

A.3 Dataset statistics and splits

We refer to the dataset as RPF-1M because it contains over one million dense process-feedback records after filtering; the current release contains 1.72M timestamped annotation anchors. RPF-1M is built from instruction-conditioned robot rollouts with available camera streams. When synchronized multi-view observations are available, all views are retained; otherwise the rollout is used as a single-view example. We retain the original RoboReward data splits at the trajectory level. Test trajectories are excluded from training PrefixCritic, RoboCritic, and the reward teacher; frames from the same rollout do not appear in both training and evaluation splits. Table 9 summarizes dataset composition by source over train, validation, and test splits.

A.4 Dense Labeling Pipeline

RPF-1M provides dense process feedback for instruction-conditioned robot rollouts. For each rollout, we estimate reward scores from sampled observation prefixes and annotate physical states over semantic execution intervals. Textual feedback is annotated over segments defined using the reward trace, with a constant score within each feedback segment. The annotations are then aligned with training output queries at 0.5 Hz, combining the reward score, state description, and segment-level feedback at each query timestamp.

RPF-1M process-label generation.

1. **Dense reward labeling.** We train a multi-view variant of RoboReward on the RoboReward dataset, initializing it from Qwen3-VL-4B, and use it as the reward teacher. We treat each sampled timestep as a temporary final frame and score the corresponding instruction-conditioned video prefix with this teacher. The teacher predicts a dense progress score on the same 1–5 scale as the terminal reward, using all synchronized camera views when available and the original single view otherwise.
2. **Reward smoothing.** We combine raw prefix scores with the rollout-level terminal label y_T. A debounce filter accepts a score transition only if it persists for at least three consecutive sampled queries; otherwise isolated samples are replaced by the previous stable score. Debounce removes local noise and does not impose monotonicity, so decreases after regressions and increases after recovery are kept, including traces that exceed y_T and later fall (e.g. $1{\to}2{\to}3{\to}4{\to}2$ when $y_T = 2$). Terminal consistency constrains only the last query: $s_T = y_T$. Episodes whose final segment disagrees with y_T are filtered out. Successful episodes ($y_T = 5$) receive score 5 only after a visible success event. We retain raw prefix scores for analysis and use the smoothed trace as the training label.
3. **State tags and captions.** We send the instruction, video, and smoothed dense reward trace to Seed-Pro 2.0. The VLM segments the video into semantic execution stages based on visible motion, contact, object-state, target-relation, and progress changes. Each segment receives one shared structured state tag and one short free-form caption.
4. **Segment-level textual feedback.** We segment the rollout using the smoothed reward trace so that the reward score is constant within each feedback segment. Seed-Pro 2.0 generates a score explanation, the conditions needed to increase the score or complete the task, and a next-step or corrective instruction, grounded in the execution state and task instruction. Each segment shares one feedback annotation; reward increases and decreases are reflected in the segment boundaries.
5. **Alignment to training queries.** Model feedback outputs are scheduled at 0.5 Hz, or every two seconds, in the training sequence. Each output timestamp receives the score and textual feedback of its containing feedback segment, together with the state tags and caption aligned to that timestamp. Multiple queries in the same feedback segment therefore share the score and textual feedback.

For the held-out test subset, human annotators review the video, instruction, reward trace, tags, captions, and feedback. Ambiguous examples are corrected when the intended label is visually determined, and removed when the state or outcome cannot be inferred. Table 10 compares automatic labels to these human-polished records.

Table 10: Automatic vs. human-polished labels on the test split (Acc↑ / MAE↓).

	Reward (raw)	Reward (smoothed)	Tags	Caption	Feedback
Acc / MAE	0.58 / 0.52	0.71 / 0.37	0.79 / –	0.72 / –	0.76 / –

A.5 State and language evaluation protocol

Language evaluation uses 500 human-reviewed RPF-1M test samples. Soft Tag evaluates four fields, whereas the strict exact-match analysis in Appendix C.3 additionally reports contact.

Soft tag matching. For each query, we compare the predicted and reference `phase`, `motion`, `gripper`, and `obj` (the `object` field in our schema). A missing reference value excludes that query from the corresponding field's denominator. We lowercase labels and canonicalize aliases, including `place`/`release` to `deposit`, `closed` to `closed_empty`, `holding` to `holding_object`, and manipulation verbs to `manipulate`, as appropriate to the field. A categorical field counts as a match if the normalized labels are identical, adjacent in the corresponding ordered sequence below, or members of the same synonym group. Each comparison is binary; "soft" refers to the relaxed matching criterion rather than fractional credit per field.

- **Phase:** `idle`, `approach`, `contact`, `grasp`, `lift`, `transport`, `align`, `deposit`, `retract`. The label `manipulate` also matches phases from `contact` through `deposit`, and `recover` matches `retract`.
- **Motion:** `none`, `grasp`, `lift`, `carry`, `place`, `release`. Synonym groups include {`lift`, `carry`}, {`place`, `release`}, and {`push`, `pull`, `slide`}.
- **Gripper:** `open`, `closing`, `grasping`, `holding_object`, `opening`, `releasing`, `closed_empty`. Synonym groups include {`closing`, `grasping`} and {`opening`, `releasing`}.

Object labels match on exact normalized string equality or a token-set Jaccard similarity of at least $1/3$. We compute each field's accuracy over its valid references and report the unweighted mean of the four field accuracies. Contact is not included in this Soft Tag aggregate.

VLM judge and query selection. We use GPT-5.6-luna at temperature 0, distinct from both the evaluated GPT-5 mini baseline and the Seed annotation model. We select up to six queries per episode, prioritizing phase boundaries and score changes and filling the remaining slots with evenly spaced queries. For each query, the judge receives the task, reference tags and caption, predicted caption and actionable feedback, and two frames around the query timestamp. Visual evidence takes precedence over the reference caption when they disagree. The judge returns only JSON, using the following criteria:

- **Caption grounding (0, 0.5, or 1):** Score 1 when the caption is supported by the current frames, without future events or an incorrect object or action. Score 0.5 when the main predicate is correct but a detail is incorrect or missing. Score 0 when the main object, action, or contact is incorrect or hallucinated.
- **Feedback usefulness (0, 0.5, or 1):** Score 1 when the advice is high-level, executable, and consistent with the current state. Score 0.5 when the direction is correct but vague or only partly consistent. Score 0 when the advice contradicts the state, specifies an incorrect action, or is not executable.
- **Temporal stability:** Evaluate only adjacent judged queries with unchanged reference phase. Mark a pair stable when the new advice does not negate the preceding advice; refinements are allowed. Report the fraction of eligible pairs marked stable.

B Implementation and Evaluation Protocol

B.1 PrefixCritic training and multi-view processing

PrefixCritic is a non-streaming progress-score baseline trained on the same RPF-1M trajectories and query timestamps as RoboCritic, using only the score field as its supervision target. It predicts a single ordinal score during both training and evaluation. We initialize it directly from the original Qwen3-VL-4B checkpoint, freeze the vision encoder, and update the language model and visual merger.

At query timestep t, PrefixCritic selects eight approximately uniformly spaced timesteps from the observed video prefix, always including the current timestep. If fewer than eight timesteps are available, all are used. All available synchronized camera views are included at each selected timestep. Each query is encoded independently, without a KV cache carried over from previous queries or historical feedback as input.

The multi-view prefix prompt follows a time-major format and uses the rubric in Table 7:

PrefixCritic prompt template.

```
Given the task, assign a discrete progress score reward (1,2,3,4,5)
for the robot in the video in the format: ANSWER: <score>
Rubric for end-of-episode progress
(judge only the final state without time limits)
(see the process-score rubric in Appendix A.1).
Task: {task_description}

The following recording has {N} synchronized camera views
observing the same scene.

[Time t_1]
View 1: <|vision_start|><vision_tokens><|vision_end|>
View 2: <|vision_start|><vision_tokens><|vision_end|>
...
View N: <|vision_start|><vision_tokens><|vision_end|>

[Time t_2]
View 1: <|vision_start|><vision_tokens><|vision_end|>
View 2: <|vision_start|><vision_tokens><|vision_end|>
...
View N: <|vision_start|><vision_tokens><|vision_end|>

...

Expected output:
ANSWER: <1,2,3,4,5>
```

The output follows the format `ANSWER: <score>` and is parsed for MAE and VOC evaluation.

B.2 Multi-view serialization and positional encoding

We use the same multi-view serialization for PrefixCritic and RoboCritic. Each sampled timestep is represented as a group of synchronized camera views. The serialization is time-major: observations are ordered by timestep, and views within the same timestep are listed as indexed image blocks. The view order is not assumed to have semantic meaning, and the number of views may vary across datasets and rollouts.

We retain the backbone's original positional encoding without modification. To distinguish camera views, we prepend a view tag to each image block and serialize observations in temporal order, grouping all available views at each timestep. View identity is indicated by these tags rather than by an additional positional-encoding dimension. The same format applies to single-view observations.

B.3 RoboCritic streaming implementation

RoboCritic uses the same visual-token format as PrefixCritic, but replaces independent prefix rescoring with bounded streaming memory. At query timestep t, the active context is

$$C_t = [P; V_t; H_t], \tag{7}$$

where P is a persistent sink, V_t is a recent visual memory, and H_t is a process-state memory.

The persistent sink P contains the feedback schema, the task instruction, and the first observation. These tokens are never evicted and provide a stable reference for the task, scene layout, and initial object configuration. The visual memory V_t is a FIFO queue containing the most recent K observation token blocks. Unless otherwise noted, we use $K = 8$ visual timesteps. Each observation block may contain one or more synchronized camera views serialized with the format in Appendix B.2. The feedback history H_t, also called process-state memory, is a FIFO queue containing the most recent M feedback records,

$$H_t = \{z_{t-M}, \ldots, z_{t-1}\}, \tag{8}$$

where each record contains the predicted reward, state description, and actionable feedback. We use $M = 32$ previous feedback records by default.

At each query, RoboCritic decodes the next feedback record from the active context:

$$z_t = (s_t, d_t, g_t) = f_\theta(C_t). \tag{9}$$

The generated record is then appended to H_t and can be used by later queries. At test time, H_t contains model-generated feedback records rather than ground-truth annotations, matching the deployment setting. Visual blocks and feedback records exceeding the fixed budgets are evicted.

The streaming prompt follows the same time-major multi-view format and uses the rubric in Table 7. The model directly outputs a JSON object with three top-level fields: `reward`, `state_description`, and `actionable_feedback`. The latter contains the score explanation, conditions for further progress or completion, and the next-step instruction. The full template is shown on the next page.

RoboCritic prompt template.

```
Given the task and streaming observations, assign the discrete
progress score reward (1,2,3,4,5)
for the latest Current observation.
Use only visible evidence up to that observation.
Do not assume future success.
Text-only historical steps may provide prior context, but they are
not current visual evidence.

Rubric for end-of-episode progress (judge only the final state
without time limits):
(see the process-score rubric in Appendix A.1).

Task: {task_description}

The following observations have {N} synchronized camera views
observing the same scene.

Initial observation:
[Time t_0]
View 1: <|vision_start|><vision_tokens><|vision_end|>
View 2: <|vision_start|><vision_tokens><|vision_end|>
...
View N: <|vision_start|><vision_tokens><|vision_end|>

Recent observations:
[Time t_{t-k}]
View 1: <|vision_start|><vision_tokens><|vision_end|>
View 2: <|vision_start|><vision_tokens><|vision_end|>
...
View N: <|vision_start|><vision_tokens><|vision_end|>

[Time t]
View 1: <|vision_start|><vision_tokens><|vision_end|>
View 2: <|vision_start|><vision_tokens><|vision_end|>
...
View N: <|vision_start|><vision_tokens><|vision_end|>

Previous feedback records:
{reward, state_description, actionable_feedback}
...

Tag choices:
phase options: approach, align, contact, deposit, grasp, idle,
lift, manipulate, other, recover, retract, transport
motion options: carry, close, fold, grasp, insert, lift, none,
open, other, place, pour, press, pull, push, release, remove,
rotate, slide, stack, sweep, unfold, unstack, wipe
gripper options: closed_empty, closing, grasping, holding_object,
not_visible, occluded, open, opening, releasing, unclear
contact options: grasped, held, near, none,
occluded, released, touching, unclear
object: short visible interaction object name,
or null if no object is identifiable

Output one JSON object only, without Markdown or extra text.
Use an integer reward from 1 to 5 (3 below is illustrative).
Use JSON null for object when no object is identifiable.
{
  "reward": 3,
  "state_description": {
    "tags": {"phase": "<phase>", "motion": "<motion>",
             "gripper": "<gripper>", "contact": "<contact>",
             "object": "<object>"},
    "caption": "<short grounded caption>"
  },
  "actionable_feedback": "<explanation>; <conditions>; <instruction>"
}
```

B.4 TRAINING DETAILS

Both PrefixCritic and RoboCritic are initialized directly from the original Qwen3-VL-4B checkpoint. The vision encoder is frozen, while the language model and visual merger are updated with supervised finetuning. Both models use the same RPF-1M trajectories and query timestamps. PrefixCritic is supervised only on the progress-score response. RoboCritic targets are serialized as reward score, structured state tags and caption, and textual feedback containing the score explanation, conditions for further progress, and next-step instruction. Prompt tokens are masked from the loss, and the autoregressive loss is applied only to each model's target response. Training queries are placed at 0.5 Hz and aligned with the corresponding segment annotations. PrefixCritic processes each sampled prefix independently without feedback history. RoboCritic uses annotated previous feedback during training and generated feedback during inference. The training output frequency specifies temporal supervision spacing rather than measured decoding throughput.

Table 11: Training hyperparameters for the streaming process critic.

Hyperparameter	**Value**
Backbone	Qwen3-VL-4B
Initialization	Original Qwen3-VL-4B checkpoint
Trainable modules	language model + visual merger
Visual budget K	8 visual timesteps
Feedback history M	32 records
Training feedback rate	0.5 Hz
Image resolution	dynamic resize, 3,136–147,456 pixels
Learning rate	3×10^{-6}
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Weight decay	0.05
Learning-rate schedule	cosine decay
Training epochs / steps	10 epochs / 8,000 optimizer steps

B.5 BASELINE EVALUATION PROTOCOL

Reward predictions are evaluated on the same 1–5 scale. Continuous progress predictions are mapped to this scale using validation-set calibration. General VLM baselines receive the same reward rubric and visual budget whenever supported. LocalWindow is trained to predict all three feedback fields. Matching the number of visual timesteps does not match total context: RoboCritic additionally retains generated feedback records. Table 12 describes context access rather than measured computational cost.

Time Proxy uses the eventual rollout duration T to interpolate linearly from 1 to 5. It provides a non-visual reference for time-correlated progress but cannot represent stalls or regressions. It is a retrospective diagnostic and cannot be treated as a causal deployment baseline when the final duration is unavailable online.

Table 12: Baseline inputs and context reuse. Visual timesteps may contain multiple camera views; K and M are not token counts or runtime measurements.

Method	**Input at query time**	**Context reuse**
Time Proxy	$1+4t/T$ from normalized timestep, using the eventual rollout duration T.	–
Robometer	Dense progress model evaluated on rollout prefixes.	Prefix input
LocalWindow	Task instruction and the most recent $K = 8$ visual timesteps, without an additional initial observation or the full persistent sink.	Bounded cache
PrefixCritic	K timesteps sampled approximately uniformly from the current prefix, with all available views at each sampled timestep.	Re-encoded
RoboCritic	Persistent sink, recent K visual timesteps, and M process-state records.	Bounded cache
General VLMs	Same task instruction, reward rubric, and visual budget when supported.	Re-encoded

B.6 METRICS AND UNCERTAINTY

If a progress-score response cannot be parsed, we retain the preceding score for the current query rather than discard the query. If parsing fails at the first query, the fallback score is 0. This fallback is included in evaluation and lies outside the normal 1–5 scoring rubric. For reward prediction, we report mean absolute error on the 1–5 scale:

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N} |\hat{s}_n - s_n|. \tag{10}$$

For terminal-query evaluation (Table 3), all methods use the same complete set of 2,831 test trajectories (1,000 from RoboArena and 1,831 from OXE), with no trajectory exclusions. Overall MAE averages the per-trajectory absolute errors directly, equivalent to weighting the two source-specific MAEs by their trajectory counts.

For streaming evaluation, we also report value-order correlation (VOC), which measures whether predicted rewards preserve the ordering of process states within the same rollout. For a trajectory τ with query set Q_τ, we compute

$$\mathrm{VOC}(\tau) = \rho_S\left(\{\hat{s}_t\}_{t\in Q_\tau}, \{s_t\}_{t\in Q_\tau}\right), \tag{11}$$

where ρ_S is Spearman rank correlation with standard tie handling. We average VOC only over trajectories for which both the ground-truth and predicted score sequences contain at least two distinct values. Trajectories with constant predictions are excluded from VOC, but remain in MAE evaluation. Consequently, the set of trajectories contributing to VOC can differ across methods. The result tables do not include confidence intervals or quantify variability across model-training or RL seeds.

C ABLATIONS AND ADDITIONAL ANALYSIS

We provide additional analysis of the main design choices in RoboCritic. The experiments use the same backbone, training split, and evaluation protocol as the main results unless otherwise specified.

C.1 STATE DESCRIPTIONS AND LANGUAGE GUIDANCE

Table 6 compares score-only (0.441 MAE / 0.732 VOC), score+state (0.413 / 0.801), and the full schema (0.404 / 0.821). The score-only variant predicts and retains progress scores; score+state additionally includes state tags and captions; the full variant also includes textual guidance. Training uses annotated histories, while evaluation uses each model's generated histories. The history budget fixes the number of retained records, with their contents determined by the variant's output schema.

C.2 STREAMING MEMORY COMPONENTS

We first ablate the components of the streaming context. RoboCritic uses three types of memory: a persistent sink that stores the schema, task instruction, and initial observation; a recent visual memory that stores the latest K visual timesteps; and a process-state memory that stores the latest M feedback records. Table 5(a) in the main text removes each component while keeping the rest of the training and inference pipeline fixed. The persistent sink provides a stable reference for the task and scene layout, the recent visual memory captures local contact and motion evidence, and the process-state memory carries compact long-horizon execution history across queries.

C.3 STATE-TAG PREDICTION

In addition to scalar rewards, RoboCritic predicts five structured state fields: phase, motion, gripper, contact, and object. We evaluate tag prediction against the human-cleaned RPF-1M annotations. For categorical fields, we report accuracy after normalizing tag strings to the predefined vocabulary. For object fields, we report normalized exact-match accuracy after lowercasing and removing articles and punctuation. These strict exact-match scores complement the soft tag accuracy in the main text, which uses relaxed matching for four fields, excluding contact (Appendix A.5).

Table 13: State-tag strict exact-match accuracy on the human-cleaned evaluation split.

Field	**Accuracy** ↑
Phase	40.0%
Motion	48.4%
Gripper state	55.8%
Contact	56.2%
Object	51.6%
Average	50.4%

C.4 Memory budget

We next vary the visual-memory size K and feedback-memory size M. The default setting in the main experiments is $K = 8$ visual timesteps and $M = 32$ previous feedback records. The visual-memory sweep varies K with $M = 32$; the feedback-memory sweep varies M with $K = 8$.

Table 14: Effect of memory budget. Left: visual-memory sweep with $M = 32$. Right: feedback-memory sweep with $K = 8$.

K	**MAE** ↓	**VOC** ↑
2	0.478	0.751
4	0.432	0.784
8	0.404	0.821

M	**MAE** ↓	**VOC** ↑
0	0.432	0.781
16	0.416	0.805
32	0.404	0.821

C.5 Failure-state subset

We also evaluate on a subset of states involving execution failures or recovery. We include query states whose human-cleaned annotations contain failure-related tags, such as dropped objects, missed grasps, wrong-object interactions, stalled execution, unstable placement, or incomplete insertion. This subset focuses on states where elapsed time is a poor proxy for task progress. Table 5(b) in the main text reports the corresponding MAE and VOC.

C.6 Scope of feedback-history ablations

Table 5(a) compares full feedback history with no feedback history. Table 6 additionally varies the fields used jointly as supervision and history. Neither comparison isolates individual history fields under fixed supervision; neither compares feedback history against additional visual observations under a matched total context-token budget.

C.7 Qualitative traces

Table 15 is a schematic illustration of the intended feedback behavior across a drop. It illustrates that an intermediate progress score may exceed the terminal score and that guidance may change after a regression. These schematic entries do not report predictions or human–model comparisons.

Table 15: Illustrative progress trace and desired guidance, not an empirical result. The terminal score is $y_T = 2$.

Query	Ref. s	Auto s	Time Proxy	Desired feedback
Approach	2	2	1.8	Approach the towel.
Grasp	3	3	2.6	Lift the object.
Near target	4	4	3.4	Lower and release at the target.
Drop	2	2	4.2	Re-grasp the dropped object.
Terminal	2	2	5.0	Re-grasp; do not proceed to place.

D Downstream Setup

D.1 Real-robot RL setup and reward shaping

Real-robot experiments use a single-arm AgileX Piper robot with external and wrist RealSense camera observations. We collect 25 demonstrations per task to train the initial $\pi_{0.5}$-based behavior-cloned policy, then fine-tune the same initial policy on pick-and-place and insertion tasks using RL-Token (Xu et al., 2026a). All reward-based fine-tuning methods share the RL procedure and a training budget of 50 rollouts per task. Reward queries are made at 2 Hz (once every 0.5 seconds). Evaluation uses 20 trials per method and task, separate from the training rollouts.

RoboCritic is frozen during policy fine-tuning. At query timesteps, we convert its ordinal score into dense shaping via

$$r_t^{\mathrm{RC}} = \hat{s}_t - \hat{s}_{t-1}. \tag{12}$$

Human Reward uses manually annotated progress scores on the same 1–5 scale. We apply the same adjacent-score differencing to these annotations as to RoboCritic predictions. The score categories are ordinal; taking Δs assigns equal numerical increments to adjacent categories as a practical shaping choice. The experiments do not compare this shaping with terminal-only rewards from the same model.

D.2 VLA Execution Guidance

We evaluate guidance on the same real-robot pick-and-place and insertion tasks as the RL experiments. All guidance methods use the same VLA checkpoint obtained after RL fine-tuning with RoboCritic rewards, with its parameters frozen throughout guidance evaluation. Each method is evaluated on 20 trials per task, for 40 trials in total. Guidance SR is computed by pooling successes across the two tasks. The results in Table 4 correspond to 24/40 successful trials for Qwen3-VL-4B (60%), 30/40 for GPT-5 mini (75%), and 34/40 for RoboCritic (85%). Guidance results for RoboReward and Robometer are unreported. The adjacent Recall, Precision, and F1 columns instead measure completion classification on RoboArena videos.

Guidance is queried at 0.5 Hz. At each update, the next-step or corrective instruction is extracted by parsing the model's JSON output and passed to the frozen VLA through its language interface while retaining the original task. If the JSON output is invalid, the previously active instruction is retained. The VLA executes for two seconds between guidance updates. Each trial ends when the reward and feedback indicate completion or when the two-minute time limit is reached.

The model's completion indication triggers stopping but does not itself count as a successful trial. A human evaluator independently checks whether the robot has actually completed the original task. Guidance SR reports the fraction of all 40 trials in which task completion is verified within the time limit.